Verification and Validation in Product-Line Software Engineering
(*One-page abstract submitted to the Software Technology Conference*)

Edward A. Addy
NASA/WVU Software Research Laboratory

The implementation of product-line software engineering not only introduces new activities to the software development process, such as domain analysis and domain modeling, it also impacts other aspects of software engineering. This presentation will discuss enhancements to the Verification and Validation (V&V) process necessary to adapt V&V to product-line software engineering.

V&V methods are used to increase the level of assurance of critical software, particularly that of safety-critical and mission-critical software. Software V&V is a systems engineering discipline that evaluates software in a systems context, and is used as part of a systematic risk mitigation strategy. The V&V methodology has been used in concert with various software development paradigms, but always in the context of developing a specific application system.

In order to bring the effectiveness of V&V to bear within a product-line software development process, V&V must be incorporated within the domain engineering process. Failure to incorporate V&V within domain engineering will result in higher development and maintenance costs due to losing the opportunity to discover problems in early stages of development and having to correct problems in multiple systems already in operation. Also, the same V&V activities will have to be performed for each application system that performs critical functions.

Much work has been done in the area of component certification (also called evaluation, assessment, or qualification). Ensuring that software components are reliable with respect to their specifications is necessary but not sufficient to show that the final system meets the needs of the user. Component evaluation is but one part of an overall V&V effort within product-line software engineering, analogous to code evaluation in V&V of an application system. In addition to component evaluation, V&V also considers the domain model and the generic architecture, along with the connections between domain artifacts and application system artifacts.

A working group at the 1996 Reuse Workshop developed a framework for performing V&V within reuse-based software engineering. This framework is described in more detail in an article appearing in the Annals of Software Engineering, Special Issue on Software Reuse, 1998. Domain-level V&V tasks are performed to ensure that domain products fulfill the requirements established during earlier phases of domain engineering. Transition-level tasks provide assurance that an application artifact correctly implements the corresponding domain artifact. Traditional application-level V&V tasks ensure the application products fulfill the requirements established during previous application life-cycle phases. The domain and the generic architecture serve as the context for evaluating software components in a product-line environment

Although not shown as a specific task for any particular phase of the life-cycle, criticality analysis is an integral part of V&V. Criticality analysis is performed in V&V of application development in order to allocate resources to the most important (i.e., critical) areas of the software. This assessment of criticality and the ensuing determination of the level of intensity for V&V tasks are crucial also within product-line software engineering. Criticality analysis and risk assessment must consider the use of components in multiple application systems and the impact of the generic architecture on meeting current and future requirements. The Component Verification, Validation and Certification Working Group at WISR 8 listed four considerations that should be used in determining the level of V&V of reusable components:

* Span of application – the number of components or systems that depend on the component
* Criticality – potential impact due to a fault in the component
* Marketability – degree to which a component would be more likely to be reused by a third party
* Lifetime – length of time that a component will be used